

# Learning to Ask: Conversational Product Search via Representation Learning

JIE ZOU, University of Amsterdam, The Netherlands

JIMMY XIANGJI HUANG, York University, Canada

ZHAOCHUN REN, Shandong University, China

EVANGELOS KANOULAS, University of Amsterdam, The Netherlands

Online shopping platforms, such as Amazon and AliExpress, are increasingly prevalent in the society, helping customers purchase products conveniently. With recent progress on natural language processing, researchers and practitioners shift their focus from traditional product search to conversational product search. Conversational product search enables user-machine conversations and through them collects explicit user feedback that allows to actively clarify the users' product preferences. Therefore, prospective research on an intelligent shopping assistant via conversations is indispensable. Existing publications on conversational product search either model conversations independently from users, queries, and products or lead to a vocabulary mismatch. In this work, we propose a new conversational product search model, ConvPS, to assist users to locate desirable items. The model is first trained to jointly learn the semantic representations of user, query, item, and conversation via a unified generative framework. After learning these representations, they are integrated to retrieve the target items in the latent semantic space. Meanwhile, we propose a set of greedy and explore-exploit strategies to learn to ask the user a sequence of high-performance questions for conversations. Our proposed ConvPS model can naturally integrate the representation learning of the user, query, item, and conversation into a unified generative framework, which provides a promising avenue for constructing accurate and robust conversational product search systems that are flexible and adaptive. Experimental results demonstrate that our ConvPS model significantly outperforms state-of-the-art baselines.

**CCS Concepts:** • **Information systems** → **Retrieval models and ranking; Specialized information retrieval; Users and interactive retrieval.**

**Additional Key Words and Phrases:** Conversational Product Search; Learning to Ask; Representation Learning

## ACM Reference Format:

Jie Zou, Jimmy Xiangji Huang, Zhaochun Ren, and Evangelos Kanoulas. 2022. Learning to Ask: Conversational Product Search via Representation Learning. *ACM Transactions on Information Systems* 1, 1 (November 2022), 27 pages. <https://doi.org/XXX/XXX.XXX>

## 1 INTRODUCTION

The rapid growth of digital markets and the widespread use of e-commerce platforms have led to a surge of research activity in the field of e-shopping [57, 78, 85]. A key functionality of e-shopping

---

Jie Zou, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands, j.zou@uva.nl; Jimmy Xiangji Huang (corresponding author), York University, 4700 Keele St, Toronto, Canada, jhuang@yorku.ca; Zhaochun Ren (corresponding author), Shandong University, 72 Binhai Highway, Qingdao, Shandong Province, China, zhaochun.ren@sdu.edu.cn; Evangelos Kanoulas, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands, e.kanoulas@uva.nl.

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.

1046-8188/2022/11-ART \$15.00

<https://doi.org/XXX/XXX.XXX>

platforms is product search, which aims to support users to effectively locate products that they wish to buy [57].

Traditional product search systems usually require customers to formulate queries and browse through the resulted products to locate the target items (*i.e.*, items they finally purchase). Searching and browsing turn out to be inefficient. Often this is due to the fact that the customers' descriptions of products do not match the vendors' descriptions [41]. The experience may become worse for mobile users since scanning a long list of products is impractical on limited bandwidth interfaces [74]. This leads to an increasing concern as mobile users continue to rapidly expand around the world. Fortunately, the rise of intelligent assistants (*e.g.*, Google Now, Apple Siri, and Microsoft Cortana) and dialogue systems provide new interaction modes between humans and systems through conversations. This lays the groundwork for a conversation-based intelligent assistant for product search, with the aim to alleviate the burden of reformulating queries and browsing through dozens of products, as well as offer a more natural way to dictate preferences and product characteristics. Compared to traditional product search, conversational product search systems interact with users by natural language and collect explicit feedback from users to clarify users' needs, leading to a better understanding of users' dynamic preferences.

Existing publications on product search mostly focus on traditional product search, using exact term matching methods [50] or semantic matching functions [3, 63] to match the query with the product directly or in the latent space. Conversational product search, on the other hand, remains a challenging open problem. Zhang et al. [78] presented the first conversational model for product search, in which a unified approach was proposed for conversational product search and recommendation by asking questions over aspect-value pairs extracted from user reviews. Similarly, Bi et al. [9] also collected user feedback on the aspect-value pairs while they proposed a paradigm for conversational product search based on negative feedback. Instead of item aspects, [85] asked questions based on extracted informative terms and proposed a question-based Bayesian product search model. Although these aforementioned approaches demonstrate their success to some extent, there are still some limitations. Bi et al. [9] focused on learning from negative feedback from shown items and model conversations *fully independently*. [85] updated user preferences and selected products on the basis of lexical overlap between terms in clarifying questions and product descriptions, leading to a *hard-matching problem*, *i.e.*, ignoring the fact that a term may not appear in the description of a product but a synonym term or passage may make the product still relevant. Zhang et al. [78] treated users, queries, products, and conversations in the same way by creating a concatenation of word embeddings to express user preferences, which *neglects the hierarchical structure* of users, queries, products, and conversations, and may suffer from the *vocabulary mismatch*.

To relax the above limitations, in this work we propose a **Conversational Product Search** model, ConvPS, to assist users in reaching their target items interactively. We carefully design our model as a unified generative model, which integrates representation learning of user, query, item, and conversation into a unified framework (rather than model them fully independently). To this end, the hard-matching problem, hierarchical structure problem, and vocabulary mismatch problem are also effectively alleviated by conducting product retrieval via soft-matching based on the jointly learned user, query, item, and conversation representations in the learned hierarchical latent semantic space. The probability of observed user-query-conversation-item quadruples can be deduced directly from their distributed representations, making the whole framework explainable and extendable. Our ConvPS model maintains a separate conversation representation and then combines it into the user representation and query representation to update the ranked list of items. Specifically, our ConvPS model (1) constructs a question pool based on slot-value pairs (aspect-value) following [9, 78], (2) learns the representation of users, queries, items, and conversations in terms of slot-value pairs, (3)

learns to ask a sequence of high-performance questions (slots) to the user based on a greedy or an explore-exploit strategy, (4) collects user feedback and enriches the user query with conversation representations, and (5) retrieves target items by using the user representation and the enriched query representation.

The major contribution of this paper is three-fold:

- (1) First, we propose a new conversational product search model, ConvPS, that integrates the representation learning of user, query, item, and conversation into a unified generative framework, and thus retrieving the relevant items based on these jointly learned representations in the latent semantic space.
- (2) Second, we propose four greedy and explore-exploit learning to ask strategies to select a sequence of high-performance questions to ask, perform analysis to compare these learning to ask strategies and validate their effectiveness.
- (3) Third, we propose a framework that can effectively incorporate two independent modules: the representation learning module for retrieving items and learning to ask module for selecting questions to ask.

Our experiments on the product search datasets demonstrate that our ConvPS model significantly improves the performance of product search compared to state-of-the-art baselines.

The rest of this paper is organized as follows. In Section 2, we discuss the related work. Section 3 introduces the details of our approach. Section 4 describes the experimental setup, research questions, experimental results and corresponding analysis, while Section 5 concludes the paper.

## 2 RELATED WORK

In this section, we summarize the related work which falls into three categories: product search, learning to ask, and conversational search. There is a large number of studies on the aforementioned topics. Here we only review work closely related to our research presented in this article.

### 2.1 Product Search

Compared with ad-hoc retrieval tasks (e.g., Web search) for finding relevant documents [27, 30], product search focuses on locating not only relevant products but also relevant products a user is willing to purchase [3, 71], making the task more challenging. Given a user query, multiple products could be relevant but only a few are actually purchased. Further, often product descriptions come with metadata (e.g., brand name, types, and categories), leading to a semi-structured information space. Considerable work has been done on product search based on structured product information [43]. Despite the effectiveness structured knowledge can offer, Vandic et al. [65, 66] found free-form user queries do not make use of structured knowledge available on product information pages. Instead, there is a vocabulary gap between product specifications and search queries [19, 20], with the concepts expressed by different terms in the product description and user queries [63, 64]. To alleviate the problem of the vocabulary gap, plenty of attempts on representation learning have been introduced. For example, Duan et al. [19, 20] proposed a probabilistic mixture model based on query generation via two language models. Van Gysel et al. [63] presented a latent vector space model to learn representations of queries and products, so that they can be mapped to the same latent space to calculate the similarity between them. Then Ai et al. [3] noticed the importance of personalization in product search and thus proposed a hierarchical embedding model by incorporating user representations for personalized product search. Later, Xiao et al. [72] proposed a Dynamic Bayesian Metric Learning model to solve the problem of personalized product search under a streaming scenario. Ai et al. [2] and Guo et al. [24] then applied an attention mechanism to improve the performance of personalized product search. On the other hand, [4] focused on

the explainability and constructed an explainable retrieval model for product search. Ahuja et al. [1] proposed a multi-lingual multi-task learning framework and studied product search in a cross-lingual information retrieval context. The effectiveness of incorporating external information for product search is also explored by some work [10, 25, 69, 80]. Besides representation learning techniques, there are a variety of studies on extracting different features and feeding them into learning-to-rank models [29, 33]. Other factors such as diversity [73] and user satisfaction [12, 38] are also studied. Similar to previous publications, in this paper, we also attempt to alleviate the language gap between product-related documents (*e.g.*, product description and user reviews) and user queries based on representation learning. Different from the aforementioned publications that treat product search as a non-interactive problem, we propose a conversational product search approach, where ranking is conducted by a dynamic process, *i.e.*, the ranking in the next iteration incorporates the conversation based on the fine-grained feedback from users.

Recently, conversational product search has gained interest within the information retrieval community. Zhang et al. [78] treated conversational search and recommendation together and proposed a first conversational product search model with multi-memory network architecture. They ask questions over item aspects extracted from user reviews and collect user feedback on the corresponding values for the asked aspects. Similarly, Bi et al. [9] also constructed conversations and enquire user's explicit responses on the aspect-value pairs mined from product reviews while they highlight the value of negative feedback in conversational product search. Instead of item aspects, Zou and Kanoulas [85] queried users on extracted informative terms (typically entities) from item-related descriptions and reviews, and proposed a sequential Bayesian method based on a cross-user duet training for conversational product search. Zou et al. [87] conducted an empirical study to quantify and validate user willingness and the extent of providing correct answers to the asked questions in question-based product search systems. In this work, we follow Zhang et al. [78] and Bi et al. [9] to construct conversations based on aspect-value pairs. Different from the above publications, we jointly learn the representations of users, queries, items, and conversations via a unified generative model. Moreover, we propose four greedy and explore-exploit learning to ask strategies to select a sequence of high-quality questions to ask to the user, and validate their effectiveness.

## 2.2 Learning to Ask

The research on applications of learning to ask is broad. For example, Rao and Daumé III [53] learned to ask useful questions for clarification on community question answering. Hu et al. [28] and Chen et al. [14] investigated the optimal strategy of question selection on a 20 Questions game setup. Aliannejadi et al. [6], Hashemi et al. [26], Rosset et al. [56] and Zou and Kanoulas [86] focused on learning to ask informative and useful clarifying questions in information-seeking systems. Christakopoulou et al. [16], Ren et al. [55] and Zou et al. [84] aimed to select next questions to ask the user for conversational recommendation [88]. Ruotsalo et al. [58] focused on selecting relevant keywords to display to the user for exploratory search. Although the importance of learning to ask in different tasks, learning to ask in product search is still relatively unexplored. Zou and Kanoulas [85] learned to ask a good question based on user preferences and the rewards over question performances. Zhang et al. [78] predicted the next question to ask to the user by maximizing the probability of the next question based on the softmax output layer for probability estimation. Compared with their greedy question selection strategies, we propose a set of systematic learning to ask strategies, including both greedy and explore-exploit strategies along with a comparative analysis among them. Further, Zou and Kanoulas [85] asked questions about descriptive terms of items while we query users about aspects. Zhang et al. [78] asked questions over limited aspects, relying on the log-likelihood of probability estimation of the next aspect. Instead, we learn to

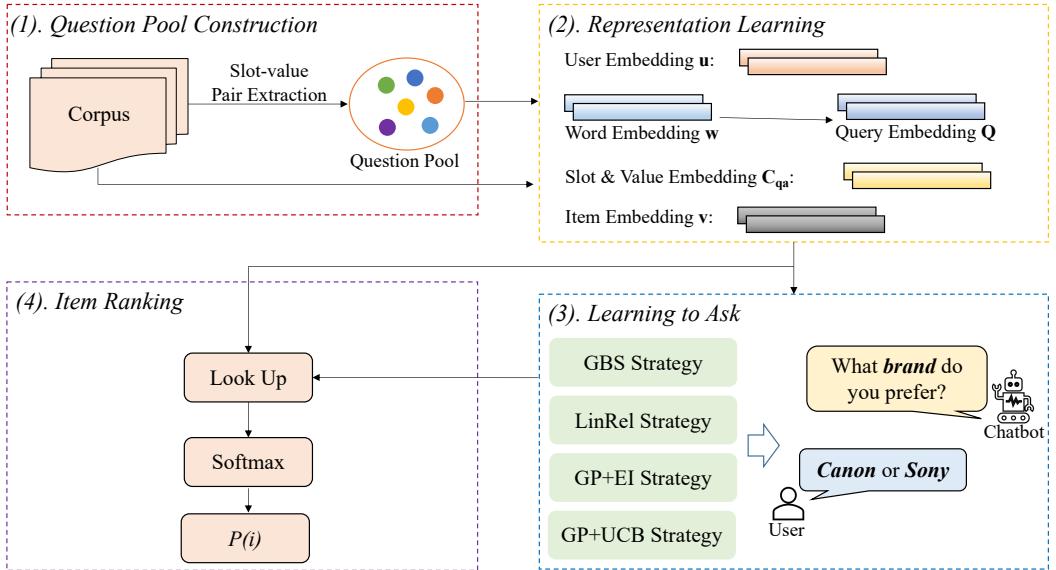


Fig. 1. The research framework of our ConvPS model. (1) We first construct a question pool via slot-value pair extraction. (2) We then learn the user embeddings, conversation embeddings in terms of slot-value embeddings, item embeddings, and word embeddings to formulate query embeddings via our generative model. (3) Afterwards, we learn to ask a sequence of questions by the four proposed strategies: GBS strategy, LinRel strategy, GP+EI strategy, and GP+UCB strategy. (4) Lastly, we generate the item ranked list by estimating the probability of items based on these learned embeddings.

ask questions over all aspects in the collection which is more challenging, based on one-hot representation in terms of aspect appearance in items.

Active learning [61] and bandit learning have also been used in the area of learning to ask. Active learning is mostly used for selecting informative items to form item-based questions to interact with users. For instance, Iovine et al. [31] investigated the effects of several item selection strategies based on active learning in the conversational recommender scenario. Christakopoulou et al. [16] deployed several active learning strategies to identify the most appropriate items to ask the user for preference elicitation. They also conducted experiments to compare the performance of active learning based question selection techniques with bandit learning based approaches. As active learning methods query for labels that provide a high amount of new information, bandit learning approaches balance the need to explore new information with a focus on exploiting what has already been learned [7]. Such an explore-exploit balance may help focus on the most relevant questions, while still considering unexplored questions with great potential of high effectiveness. Following this line, Christakopoulou and Banerjee [15] proposed a collaborative-bandit approach to tackle the explore-exploit trade-off in the problem of learning to interact with users. Zhang et al. [77] and Li et al. [42] proposed UCB methods or Thompson Sampling based model to deal with the explore-exploit trade-off in conversational recommender scenarios. In this paper, we propose both greedy and bandit learning strategies along with a comparison analysis among them in conversational product search.

Table 1. A running example. In this example user “A1TR3G8YUR8QMG” (represented by the user embedding  $\mathbf{u}$ ) issued a query “personal health foot care insert insole” (represented by the query embedding  $\mathbf{Q}$ ) and would like to locate the target item “B003FJACI8” (represented by the item embedding  $\mathbf{v}$ ). Each item is annotated by a set of slot-value pairs, which is shown in the example indicated in blue. The corresponding embeddings of user, query, item, and conversation are indicated by square brackets. For each conversation round, we select the question with the optimal score calculated by our corresponding learning to ask function to ask the user. In our example, in the first round, we selected the question “What **price** (represented by the slot embedding  $\mathbf{q}$ ) would you like?”, and received user response “I want **outstanding** (represented by the value embedding  $\mathbf{a}$ ) price”. After the first round of conversation, the rank of target item increases from 2,251 to 522.

<b>User:</b>	- $[\mathbf{u}]$	
ID: A1TR3G8YUR8QMG		
<b>Query:</b> “personal health foot care insert insole”	- $[\mathbf{Q}]$	
<b>Target item:</b>	- $[\mathbf{v}]$	
ID: B003FJACI8, Title: UGG Australia Men’s Replacement Insoles Boot Inserts. Extracted slot-value pairs: ( <b>size</b> , <b>big</b> ), ( <b>foam</b> , <b>thin</b> ), ( <b>price</b> , <b>outstanding</b> ), ( <b>shipping</b> , <b>free</b> ), ( <b>quality</b> , <b>real</b> ), ( <b>insole</b> , <b>thick</b> ) ...		
<b>Conversation:</b>	- $[\mathbf{c}_{qa}]$	
System	User	Rank of target item
		2251
What <b>price</b> $[\mathbf{q}]$ would you like?	I want <b>outstanding</b> $[\mathbf{a}]$ price.	522
What <b>quality</b> $[\mathbf{q}]$ would you like?	I want <b>real</b> $[\mathbf{a}]$ quality.	53
What <b>brand</b> $[\mathbf{q}^-]$ would you like?	No brand.	34
What <b>size</b> $[\mathbf{q}]$ would you like?	I want <b>big</b> $[\mathbf{a}]$ size.	3
...		

### 2.3 Conversational Search

Early exploration of conversational search investigated mixed-initiative systems by interacting with users via script-based conversation during a search session [8]. Recently, researchers investigated conversational search by asking clarifying questions (CQs) [56, 70, 74]. Radlinski and Craswell [52] introduced a theoretical framework for conversational search and highlighted the importance of conversational search. Among previous studies about conversational search, they can be classified into four main categories: algorithms for conversational search [26, 34, 70, 74, 86], evaluation of conversational search [35, 44], datasets for conversational search [5, 6, 17, 47, 51, 54, 75], and empirical studies for conversational search [36, 37, 62, 67, 76, 83, 87]. Compared with the aforementioned work on conversational search to retrieve documents or answers, our research focuses on product-seeking conversations.

## 3 METHODOLOGY

In this section, we introduce our proposed conversational product search model, ConvPS. A research framework based on the ConvPS model is shown in Figure 1. This framework includes four modules: (1) question pool construction; (2) representation learning; (3) learning to ask; and (4) item ranking.

Table 2. A summary of key notations in this work.

Notation	Explanation
$u_i, \mathcal{U}$	The $i$ -th user, and the set of all users, respectively
$v_j, \mathcal{V}$	The $j$ -th item, and the set of all items, respectively
$M, N$	The number of users $M =  \mathcal{U} $ , and number of items $N =  \mathcal{V} $
$Q, \mathbf{Q}$	The initial user query and the embedding of a query, respectively
$\mathbf{u}$	The embedding of a user
$\mathbf{v}$	The embedding of an item
$\mathbf{w}$	The embedding of a word
$(q, a)$	The slot-value pair
$\mathbf{c}_{qa}$	The corresponding embedding of a slot-value pair $(q, a)$
$\mathbf{x}_q$	The one-hot vector of slot $q$
$q_l, L$	The $l$ -th selected slot and the length ( <i>i.e.</i> , number of slots) of a conversation, respectively
$\mathbf{q}, \mathbf{a}$	The trained embeddings of a slot with positive feedback, and a value, respectively
$\mathbf{q}^-$	The trained embeddings of a slot with negative feedback
$\mathcal{D}_w$	The set of vocabulary of words in the corpus
$\mathcal{D}_u$	The words in textual descriptions of user $u$
$\mathcal{D}_v$	The words in textual descriptions of item $v$
$\mathcal{S}_c$	The set of slot-value pairs in the corpus
$\mathcal{S}_u$	The slot-value pairs in the historical conversations of user $u$
$\mathcal{S}_v$	The slot-value pairs in the historical conversations of item $v$
$\mathcal{S}_{uQv}$	The slot-value pairs in the conversations of a user-query-item triple

In the following, we will first describe the conversational product search problem and formally define our model, and then describe each of the four modules in detail. A running example is provided in Table 1.

### 3.1 Problem Formalization

Assume we have  $M$  users  $\mathcal{U} = \{u_1, u_2, \dots, u_M\}$  and  $N$  items  $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ . Each item  $v_j \in \mathcal{V}$  is a product in e-commerce, and each item has its textual description  $d_j$ . A search session is initiated with a query  $Q$  issued by a user  $u_i$ . During the search session, suppose we ask the user a clarifying question and the user provides an (associated) answer for the asked clarifying question, denoted as  $(q', a')$ . In the conversation for each search session, we ask the user a sequence of clarifying questions and collect a sequence of user answers for the asked clarifying questions. Suppose we asked  $L$ -round questions, the sequence of clarifying questions and answers are denoted as  $\{(q'_1, a'_1)(q'_2, a'_2), \dots, (q'_L, a'_L)\}$ . Thus, the sequence of actions in the conversation can be represented with:

$$u_i \rightarrow Q \rightarrow q'_1, a'_1, q'_2, a'_2, \dots, q'_L, a'_L, \rightarrow v_j^*, \quad (1)$$

where  $v_j^*$  is the target item the user is looking for. The goal of the system is, after collecting the  $L$ -iteration user's feedback, to show a ranked list of items that ranks the target item  $v_j^*$  on the top.

### 3.2 Question Construction

In this work, the questions in conversations are constructed based on clarification features, which are extracted or predefined from all the textual descriptions. This means we ask clarifying questions

about clarification features by using templates to generate them, similar to existing publications [9, 74, 78, 84–86]. To collect clarification features, in this work, we use slot-value pairs, and in particular a set of aspect-value pairs extracted from item descriptions and user reviews following the approach taken by Bi et al. [9] and Zhang et al. [78]. Note that one could also use other features like keywords, item categories, or facets to construct questions. We leave the investigation of natural language generation (*i.e.*, generating natural language sentences from slot-value pairs) and natural language understanding (*i.e.*, understanding natural language sentences to extract slot-value pairs) as future work. Accordingly, the system can ask a sequence of questions in the form of “what [slot/aspect] would you like?”, the users can respond with specific values corresponding to the asked slot, or “not relevant” as the negative feedback according to their information needs. We denote the extracted slot-value pair in the conversation as  $(q, a)$ . For example, (size, big) is a slot-value pair  $(q, a)$  and it can construct a question “what size would you like?”, as shown in our running example in Table 1.

### 3.3 Representation Learning

In this section, we will explain how we learn different representations, including user representations, item representations, slot & value representations, and query representations via a unified generative framework.

**3.3.1 Item Generation Model.** Following the generative process of Equation 1, we construct an item generation model. An item  $v$  is generated from a user  $u$ , the user’s initial query  $Q$ , and a sequence of clarifying questions and their associated answers  $(q', a')$  which is represented by slot-value pairs  $(q, a)$ . Inspired by the work of embedding-based generative framework [3, 4, 9, 46, 78], we use a softmax function to compute the probability of an item  $v$  conditioned by the user embedding, query embedding, and slot-value embeddings:

$$P(v|u, Q, q, a) = \frac{\exp(v \cdot (\lambda_u u + \lambda_Q Q + \lambda_c c_{qa}))}{\sum_{v' \in \mathcal{V}} \exp(v' \cdot (\lambda_u u + \lambda_Q Q + \lambda_c c_{qa}))), \quad (2)$$

where  $c_{qa}$ ,  $v$ ,  $u$ , and  $Q$  are the embeddings of  $(q, a)$ , items, users, and queries, respectively (will be introduced next).  $\lambda_u$ ,  $\lambda_Q$ , and  $\lambda_c$  are hyper-parameters that control the weight of the user, query, and slot-value pairs, respectively. Note that  $\lambda_u$  can be 0, which means the model will be degraded to a non-personalized search model.  $\lambda_c$  can also be 0, which means the model will be degraded to a non-conversational search model.

**3.3.2 Item and User Language Model.** Inspired by Ai et al. [3], we learn the representation of items and users by constructing item/user language models. While most of the latent vector space models construct item/user language models with textual description (*i.e.*, words) only [3, 9], we construct item/user language models from two perspectives, including both textual descriptions and clarification features (*i.e.*, slot-value pairs), to learn a better representation of items and users.

- When considering textual descriptions, given the item embedding  $v$  ( $v \in \mathbb{R}^t$ ) and the embedding of a word  $w$  ( $w \in \mathbb{R}^t$ ), the probability of  $w$  being generated from the item language model is defined with a softmax function on item embedding  $v$  and word embedding  $w$ :

$$P(w|v) = \frac{\exp(w \cdot v)}{\sum_{w' \in \mathcal{D}_w} \exp(w' \cdot v)}, \quad (3)$$

where  $\mathcal{D}_w$  is the set of the vocabulary of words in the textual descriptions from the corpus. Similarly, we have a user language model:

$$P(w|u) = \frac{\exp(w \cdot u)}{\sum_{w' \in \mathcal{D}_u} \exp(w' \cdot u)}. \quad (4)$$

- When considering clarification features, given the item embedding  $\mathbf{v}$  ( $\mathbf{v} \in \mathbb{R}^t$ ) and the embedding of a associated slot-value pair  $\mathbf{c}_{qa}$  ( $\mathbf{c}_{qa} \in \mathbb{R}^t$ ), the probability of  $(q, a)$  being generated from the item language model is defined with a softmax function on the item embedding  $\mathbf{v}$  and the slot-value embedding  $\mathbf{c}_{qa}$ :

$$P((q, a) | v) = \frac{\exp(\mathbf{c}_{qa} \cdot \mathbf{v})}{\sum_{(q, a)' \in \mathcal{S}_c} \exp(\mathbf{c}'_{qa} \cdot \mathbf{v})}, \quad (5)$$

where  $\mathcal{S}_c$  is the set of clarification features from the corpus. Similarly, we have a user language model:

$$P((q, a) | u) = \frac{\exp(\mathbf{c}_{qa} \cdot \mathbf{u})}{\sum_{(q, a)' \in \mathcal{S}_c} \exp(\mathbf{c}'_{qa} \cdot \mathbf{u})}. \quad (6)$$

The item and user language models allow us to make use of the sources of clarification features and textual descriptions, by connecting the embedding of clarification features and the embedding of words with user/item embeddings. This enables us to directly measure the semantic similarity between words and items/users in latent semantic space, as well as the semantic similarity between clarification features and items/users in latent semantic space.

**3.3.3 Slot-value Representation.** We utilize the informative slot-value pairs  $(q, a)$  to represent the conversation, where  $q$  represents the slots and  $a$  represents the values. In this work, we consider both positive feedback and negative feedback from users. We define the representation of slot-value pairs  $\mathbf{c}_{qa}$  for positive feedback as:

$$\mathbf{c}_{qa} = \frac{\mathbf{q} + \mathbf{a}}{2}, \quad (7)$$

where  $\mathbf{q}$  and  $\mathbf{a}$  are the embedding of the slot in a question and the value in an answer, respectively. For simplicity, we define  $\mathbf{c}_{qa}$  as the average of slot and value representations. Note that the way of combining slot and value representations for  $\mathbf{c}_{qa}$  is ad hoc and follows early approaches that pivot on the additive compositionality of embeddings (see e.g., [46].); one can define  $\mathbf{c}_{qa}$  as any other functions on top of slot and value representations, e.g., adding an adaptive weight for the slot and value representations or combining them by a nonlinear projection function [3].

Besides positive feedback, negative feedback is also a strong signal and is important to be incorporated into the model [9]. Compared with positive feedback, negative feedback is more challenging since relevant items usually have similar characteristics while the reason for an item to be non-relevant could vary. In this work, we learn a separate embedding for negative feedback. We utilize  $\mathbf{q}^-$  to represent the negative feedback on the asked slot  $q$ .<sup>1</sup> That is, we train both  $\mathbf{q}$  and  $\mathbf{q}^-$  for each  $q$ . Then we have the embedding of a clarifying question with negative feedback:

$$\mathbf{c}_{qa} = \mathbf{q}^-. \quad (8)$$

**3.3.4 Query Representation.** As there are a large number of possible queries, it is impractical to generalize offline learned query embeddings to unseen queries. That is,  $\mathbf{Q}$  must be calculated in product search at the request time. We then divide queries into words and construct query embeddings  $\mathbf{Q}$  by word embeddings  $\mathbf{w}$  based on a nonlinear projection function similar to other publications [3, 63] :

$$\mathbf{Q} = \tanh(\mathbf{W} \cdot \frac{\sum_{w \in Q} \mathbf{w}}{|Q|} + \mathbf{b}), \quad (9)$$

where  $\mathbf{W} \in \mathbb{R}^{t \times t}$  and  $\mathbf{b} \in \mathbb{R}^t$  are parameter matrices and bias vectors to be learned in the training process, respectively.  $|Q|$  is the length of the query. Again, the way to combine word embeddings to form a query embedding is ad hoc; one can define query embedding as other functions, e.g.,

---

<sup>1</sup>Remember that negative feedback on the slot represents the user finding the slot irrelevant to the item she is looking for.

aggregating and average the embeddings of query words directly [68], or sequentially input the query words into a recurrent neural network (RNN) and use the final network state as the latent query representation [49].

**3.3.5 Joint Learning Framework.** With all the components introduced previously, we can jointly learn the embeddings of queries, users, items, and slot-value pairs by maximizing the likelihood of the observed user-query-conversation-item quadruple in the training set. Let  $\mathcal{S}_{uQv}$  be the set of slot-value pairs in the conversations for the interaction of user  $u$  and item  $v$  with query  $Q$ . Let  $\mathcal{S}_u$  be the set of associated slot-value pairs of  $u$  (*i.e.*, slot-value pairs in the historical conversations for historical purchases of user  $u$ ),  $\mathcal{S}_v$  be the set of associated slot-value pairs of  $v$  (*i.e.*, slot-value pairs in the historical conversations for historical purchases of item  $v$  by all users),  $\mathcal{D}_v$  be the textual descriptions (*i.e.*, item description and reviews) of  $v$ , and  $\mathcal{D}_u$  be the textual descriptions (*i.e.*, user reviews) of  $u$ . Then, the maximization of the likelihood of observing a user-query-conversation-item quadruple with corresponding slot-value pairs and textual descriptions associated with users or items in our model can be formulated as follows:

$$\mathcal{L}(u, Q, v, \mathcal{S}_{uQv}, \mathcal{S}_v, \mathcal{S}_u, \mathcal{D}_v, \mathcal{D}_u) = \log P(u, Q, v, \mathcal{S}_{uQv}, \mathcal{S}_v, \mathcal{S}_u, \mathcal{D}_v, \mathcal{D}_u). \quad (10)$$

In our ConvPS model, words in  $\mathcal{D}_v$  and  $\mathcal{D}_u$  are generated by the language model of  $v$  and  $u$ , which are Equation 3 and Equation 4, respectively.  $\mathcal{D}_v$  is independent of  $u$ ,  $Q$ , and  $\mathcal{D}_u$  while  $\mathcal{D}_u$  is independent of  $v$ ,  $Q$ , and  $\mathcal{D}_v$ . Slot-value pairs in  $\mathcal{S}_v$  and  $\mathcal{S}_u$  are also generated by the language model of  $v$  and  $u$ , which are Equation 5 and Equation 6, respectively.  $\mathcal{S}_v$  is independent of  $u$ ,  $Q$ ,  $\mathcal{S}_{uQv}$  and  $\mathcal{S}_u$  while  $\mathcal{S}_u$  is independent of  $v$ ,  $Q$ ,  $\mathcal{S}_{uQv}$  and  $\mathcal{S}_v$ . For simplicity, we assume that  $\mathcal{S}_u$ ,  $\mathcal{S}_v$ , and  $\mathcal{S}_{uQv}$  are independent each other. We also assume that  $\mathcal{D}_v$  and  $\mathcal{D}_u$  are independent from  $\mathcal{S}_{uQv}$ ,  $\mathcal{S}_v$ , and  $\mathcal{S}_u$ . Thus, we can rewrite Equation 10 as follows:

$$\begin{aligned} \mathcal{L}(u, Q, v, \mathcal{S}_{uQv}, \mathcal{S}_v, \mathcal{S}_u, \mathcal{D}_v, \mathcal{D}_u) \propto & \sum_{(q,a) \in \mathcal{S}_{uQv}} \log P(v|u, Q, q, a) + \sum_{(q,a) \in \mathcal{S}_v} \log P((q, a)|v) \\ & + \sum_{(q,a) \in \mathcal{S}_u} \log P((q, a)|u) + \sum_{w \in \mathcal{D}_v} \log P(w|v) + \sum_{w \in \mathcal{D}_u} \log P(w|u). \end{aligned} \quad (11)$$

Therefore, the log-likelihood is actually the sum of log-likelihood for the user language model, the item language model, and the item generation model. To better learn the representations and alleviate the potential mismatch between the query and item for a given user when there are no conversations (*e.g.*, the initial item ranking before asking questions), we also add a non-conversational term for user-query-item triples, *i.e.*, a non-conversational version of Equation 2, into the loss function for optimizing:

$$P(v|u, Q) = \frac{\exp(v \cdot (\lambda_u \mathbf{u} + \lambda_Q \mathbf{Q}))}{\sum_{v' \in \mathcal{V}} \exp(v' \cdot (\lambda_u \mathbf{u} + \lambda_Q \mathbf{Q}))}. \quad (12)$$

Our ablation experiments in Section 4.4 also show better performance of involving such a non-conversational term for training. By adding it into our loss function, then we have:

$$\begin{aligned} \mathcal{L}(u, Q, v, \mathcal{S}_{uQv}, \mathcal{S}_v, \mathcal{S}_u, \mathcal{D}_v, \mathcal{D}_u) &= \log P(u, Q, v, \mathcal{S}_{uQv}, \mathcal{S}_v, \mathcal{S}_u, \mathcal{D}_v, \mathcal{D}_u) \\ &\propto \log P(v|u, Q) + \sum_{(q,a) \in \mathcal{S}_{uQv}} \log P(v|u, Q, q, a) + \sum_{(q,a) \in \mathcal{S}_v} \log P((q, a)|v) \\ &+ \sum_{(q,a) \in \mathcal{S}_u} \log P((q, a)|u) + \sum_{w \in \mathcal{D}_v} \log P(w|v) + \sum_{w \in \mathcal{D}_u} \log P(w|u). \end{aligned} \quad (13)$$

Computing the log-likelihood involving softmax function directly is impractical, since its denominator contains the sum of a large number of elements. Same as other publications [3, 39], we approximate the estimation of the denominator of softmax functions by adopting the negative sampling strategy for efficient training. In particular, we randomly sample  $\alpha$  words or slot-value pairs from the corpus according to a predefined distribution to approximate the softmax function. Therefore, the log-likelihood of the user language model and item language model with negative sampling is:

$$\begin{aligned}
\log P(v|u, Q) &\approx \log \sigma(\mathbf{v} \cdot (\lambda_u \mathbf{u} + \lambda_Q \mathbf{Q})) \\
&\quad + \alpha \cdot \mathbb{E}_{v' \sim P_v} [\log \sigma(-\mathbf{v}' \cdot (\lambda_u \mathbf{u} + \lambda_Q \mathbf{Q}))], \\
\log P(v|u, Q, q, a) &\approx \log \sigma(\mathbf{v} \cdot (\lambda_u \mathbf{u} + \lambda_Q \mathbf{Q} + \lambda_c \mathbf{c}_{qa})) \\
&\quad + \alpha \cdot \mathbb{E}_{v' \sim P_v} [\log \sigma(-\mathbf{v}' \cdot (\lambda_u \mathbf{u} + \lambda_Q \mathbf{Q} + \lambda_c \mathbf{c}_{qa}))], \\
\log P((q, a)|v) &\approx \log \sigma(\mathbf{c}_{qa} \cdot \mathbf{v}) + \alpha \cdot \mathbb{E}_{(q, a)' \sim P_c} [\log \sigma(-\mathbf{c}'_{qa} \cdot \mathbf{v})], \\
\log P((q, a)|u) &\approx \log \sigma(\mathbf{c}_{qa} \cdot \mathbf{u}) + \alpha \cdot \mathbb{E}_{(q, a)' \sim P_c} [\log \sigma(-\mathbf{c}'_{qa} \cdot \mathbf{u})], \\
\log P(w|v) &\approx \log \sigma(\mathbf{w} \cdot \mathbf{v}) + \alpha \cdot \mathbb{E}_{w' \sim P_w} [\log \sigma(-\mathbf{w}' \cdot \mathbf{v})], \\
\log P(w|u) &\approx \log \sigma(\mathbf{w} \cdot \mathbf{u}) + \alpha \cdot \mathbb{E}_{w' \sim P_w} [\log \sigma(-\mathbf{w}' \cdot \mathbf{u})],
\end{aligned} \tag{14}$$

where  $P_w$  and  $P_c$  is defined as the word distribution and slot-value pair distribution in the corpus respectively, raised to 3/4rd power [46].  $P_v$  is defined as a uniform distribution for items.

By adding L2 regularization to avoid over-fitting, we then define our final loss function:

$$\sum_{u, Q, v} \mathcal{L}(u, Q, v, \mathcal{S}_{uQv}, \mathcal{S}_v, \mathcal{S}_u, \mathcal{D}_v, \mathcal{D}_u) + \gamma \left( \sum_{u \in \mathcal{U}} \|\mathbf{u}\|_2^2 + \sum_{v \in \mathcal{V}} \|\mathbf{v}\|_2^2 + \sum_{w \in \mathcal{D}_w} \|\mathbf{w}\|_2^2 + \sum_{(q, a) \in \mathcal{S}_c} \|\mathbf{c}_{qa}\|_2^2 \right), \tag{15}$$

where  $\gamma$  is the hyper-parameters that control the weight for L2 regularization. All the embeddings are trained simultaneously in our model.

### 3.4 Learning to Ask

In this section, we aim to learn how to ask a sequence of high-performance questions. This means, (1) selecting questions to ask to obtain enough information so that the system can suggest the good matching item to the user, and (2) selecting the questions to ask to hit the target item with as few iterations as possible. In this work, we propose four learning to ask strategies and transfer them to produce search, including Generalized Binary Search (GBS) [48], LinRel bandit [7], and two Gaussian Process (GP) variants. The first one is a greedy strategy while the last three are explore-exploit strategies.

**3.4.1 GBS.** Inspired by the successful application in Zou and Kanoulas [86] and Zou et al. [84], we explore GBS for learning to ask questions in product search. GBS is a greedy strategy that selects the slot  $q_l$  which is able to best split the estimated user preferences corresponding to the items closest to two halves:

$$q_l = \arg \min_q \left| \sum_{v \in \mathcal{V}} (2\mathbb{1}\{q^v = 1\} - 1)\pi_l(v) \right|, \tag{16}$$

where  $q_l$  is the  $l$ -th chosen question (slot),  $q^v$  expresses whether the textual description of  $v$  contains the slot  $q$  or not. Specifically, if the slot  $q$  appears in the textual description of  $v$ , then  $\{q^v = 1\}$  is true and  $\mathbb{1}\{q^v = 1\}$  is equal to 1, otherwise  $\{q^v = 1\}$  is false and  $\mathbb{1}\{q^v = 1\}$  is equal to 0.  $\pi_l(v)$  is the estimated user preferences with each dimension in terms of a ranking score of  $v$ . In particular, we set  $\pi_l(v)$  to  $1/(\text{index}_v + 1)$ , where  $\text{index}_v$  is the index of item  $v$  in the  $l$ -th iteration ranked list of items.

**Algorithm 1:** Gaussian process strategy

---

**Input:** Kernel function  $\kappa(\cdot, \cdot)$ , all of slot representation  $\mathbf{x}_q$ , number of starting points  $t$ , and number of questions  $L$

**Output:** The selected slots  $(q_{t+1}, \dots, q_L)$ :  $L > (t + 1)$

- 1 Initialization for obtaining  $t$  starting points by GBS:  $\mathcal{D}_s = \{(q_1, y_1), \dots, (q_t, y_t)\}$
- 2  $l = t + 1$
- 3 **while**  $l \leq L$  **do**
- 4     Update  $\mu_l(q)$  and  $\sigma_l^2(q)$  according to Equation 19 and Equation 20 based on  $\mathcal{D}_s$
- 5     select  $q_l$  according to Equation 21 or Equation 22
- 6     Observe  $y_l \in \{-1, 1\}$  via received feedback of  $q_l$
- 7     Adding  $(q_l, y_l)$  to  $\mathcal{D}_s$ :
- 8          $\mathcal{D}_s = \mathcal{D}_s \cup \{(q_l, y_l)\}$
- 9 **end**

---

3.4.2 *LinRel.* We also explore bandit algorithms to carry out explore-exploit for learning to ask questions, where we would like to gather information about question rewards through exploration while at the same time harvest currently available question rewards through exploitation. Firstly, we adopt a contextual bandit algorithm, LinRel [7, 58], in our task. LinRel has been proven to work well in other interactive retrieval systems [23, 58]. LinRel estimates a linear regression model, which makes use of side information (clarification features) to estimate the relevance score of a slot (question). It predicts expected slot relevance and corresponding upper confidence bounds (*i.e.*, the sum of the mean and variance of the expected relevance score). To balance the exploration and exploitation, LinRel selects the slot with the highest upper confidence bounds rather than selecting the slot with the largest estimated mean of the relevance score. Intuitively, slots with high upper confidence bounds are the ones that are either already highly relevant with less uncertainty, or potentially relevant but with greater uncertainty. Assume we have a data matrix  $\mathbf{X}$  where the slots in terms of questions are rows and items are columns. We denote  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_F)^\top$  as the matrix containing  $F$  row vectors presented so far, where  $F$  is the number of slots. For any slot  $q$ ,  $\mathbf{x}_q$  ( $\mathbf{x}_q \in \mathbf{X}$ ) is a one-hot vector of items versus that slot  $q$  with  $N$  dimensions. Each dimension corresponds to a value of 0 or 1 (when a slot is contained in the corresponding item, the value is 1, otherwise 0.). The algorithm first computes a regression weight vector for the weight of exploitation (*i.e.*, collected question rewards):

$$\mathbf{h}_q = \mathbf{x}_q \cdot (\mathbf{X}^\top \mathbf{X} + \lambda_I \mathbf{I})^{-1} \mathbf{X}^\top, \quad (17)$$

where  $\mathbf{I}$  is the identity matrix, and  $\lambda_I$  is a regularization parameter.

Suppose up to iteration  $l$  we have collected  $l-1$  relevance feedback where each feedback  $r_1, \dots, r_{l-1} \in \{-1, 1\}$ , is a relevance value for a particular slot, and the vector of all feedback values received so far is denoted  $\mathbf{r} = (r_1, \dots, r_{l-1})^\top$ . In this paper, we set relevance value of positive feedback to 1 and relevance value of negative feedback to -1. Then we select the slot  $q$  that achieve maximum value to ask to the user:

$$q_l = \arg \max_q \left\{ \mathbf{h}_q \cdot \mathbf{r} + \frac{c}{2} \|\mathbf{h}_q\|^2 \right\}, \quad (18)$$

where  $\mathbf{h}_q$  is the regression weight vector for the slot  $q$  whose relevance score we are predicting,  $\|\mathbf{h}_q\|$  is the L2 norm of the regression weight vector, and the constant  $c$  is used to adjust the explore-exploit trade-off. Intuitively, the first term on the right side of Equation 18 models exploitation and the second term models exploration.

**3.4.3 Gaussian Process.** Learning to ask questions can be regarded as making sequential decisions (*i.e.*, asking questions) to optimize an unknown payoff function, which can be typically modeled as a Gaussian Process (GP). In particular, each decision (asked question) results in a stochastic reward with an initially unknown distribution, while new decisions are taken on the basis of the observations of previous question rewards. In this section, we explore learning to ask strategies modeled by GP. The key idea is to use GP models for the received feedback of asked slots, and to obtain slot estimates via maximum likelihood. In GP, we first deduce the distribution of the relevance of slots and then select the optimal slot to ask based on the relevance score of slots. That is, we predict the relevance score of as yet unexplored slots using GP models, whose regularity is captured in the kernel (covariance) function. Exploiting such regularity allows us to efficiently maximize the benefit drawn from sparse observations about the asked slots.

Consider now we have already observed relevance feedback for  $t$  slots, *i.e.*, obtained data  $\{(q_1, y_1), \dots, (q_t, y_t)\}$ , where  $y_t$  is the observed relevance of  $q_t$ . In this work,  $y_t \in \{-1, 1\}$ , *i.e.*, it is set to 1 for positive feedback received and -1 for negative feedback received. We assume the relevance of slot  $y_t$  conforms to a GP. We also assume  $y_t$  is noisy and is composed of a “true” relevance  $R(q_t)$  plus independent Gaussian noise  $e_t$ , with  $y_t = R(q_t) + e_t$ , where  $e_t$  is a predefined hyper-parameter. The GP assumption for the observed feedback  $(y_1, \dots, y_t)$  implies that the true feedback for all slots are jointly Gaussian distributed, with zero mean. Then, for a new slot, its predictive distribution for the relevance  $R(q)$  is Gaussian, with mean and variance given by

$$\mu_t(q) = \mathbf{k}_t(\mathbf{x}_q)^\top (\mathbf{K}_t + \mathbf{I})^{-1} \mathbf{y}_t, \quad (19)$$

$$\sigma_t^2(q) = \kappa(\mathbf{x}_q, \mathbf{x}_q) - \mathbf{k}_t(\mathbf{x}_q)^\top (\mathbf{K}_t + \mathbf{I})^{-1} \mathbf{k}_t(\mathbf{x}_q), \quad (20)$$

where  $\mathbf{k}_t(\mathbf{x}_q) = [\kappa(\mathbf{x}_q, \mathbf{x}_{q_1}), \dots, \kappa(\mathbf{x}_q, \mathbf{x}_{q_t})]^T$ ,  $\mathbf{K}_t$  is the positive semi-definite kernel matrix such that  $\mathbf{K}_{t,i,j} = [\kappa(\mathbf{x}_{q_i}, \mathbf{x}_{q_j})]$ , and  $\kappa(\mathbf{x}_{q_i}, \mathbf{x}_{q_j})$  is a particular kernel function for two embeddings  $\mathbf{x}_{q_i}$  and  $\mathbf{x}_{q_j}$ . Again,  $\mathbf{x}_{q_i}$  and  $\mathbf{x}_{q_j}$  are one-hot vectors of items versus the slot  $q_i$  and  $q_j$  respectively, with each dimension corresponding to a value of 0 or 1. For two slots  $q_i$  and  $q_j$ , the kernel  $\kappa(\mathbf{x}_{q_i}, \mathbf{x}_{q_j})$  represents our assumptions about how similar we expect the relevance of  $q_i$  with representation  $\mathbf{x}_{q_i}$ , as opposed to  $q_j$  with representation  $\mathbf{x}_{q_j}$  are. In this work, we use a common kernel function: RBF kernel  $\kappa(\mathbf{x}_{q_i}, \mathbf{x}_{q_j}) = \sigma^2 \exp(-\frac{\|\mathbf{x}_{q_i} - \mathbf{x}_{q_j}\|^2}{2})$ . RBF kernel is a popular kernel function used in Support Vector Machine (SVM) [60] and various kernelized learning algorithms [40].

**Choice of acquisition function.** For GP, we need an acquisition function for deciding where to sample slots. In this work, we use two acquisition functions to select slots.

- Upper Confidence Bound (UCB):

$$q_l = \arg \max_q \mu_l(q) + \beta \sigma_l(q). \quad (21)$$

This criterion selects the slot with the highest upper confidence bound on the relevance function.  $\beta$  is a tradeoff parameter to control the explore-exploit.

- Expected Improvement (EI):

$$q_l = \arg \max_q EI_l(q), \quad (22)$$

$$EI_l(q) = (\mu_l(q) - \mu_l(q_l^*)) \Phi \frac{\mu_l(q) - \mu_l(q_l^*)}{\sigma_l(q)} + \sigma_l(q) \phi \frac{\mu_l(q) - \mu_l(q_l^*)}{\sigma_l(q)}, \quad (23)$$

where  $\phi$  and  $\Phi$  are the standard normal probability density and cumulative distribution functions, respectively.  $q_l^* = \arg \max_q \mu_l(q)$ , which denotes the slot with largest posterior mean at round  $l$ . The EI value  $EI_l(q)$  measures the potential of  $q$  to improve upon the largest posterior mean  $\mu_l(q_l^*)$  at round  $l$ . EI is another popular infill criterion in GP assisted

optimization to balance exploitation and exploration of the predictive distribution model, which has been widely used [32]. It selects the slot with the largest amount of improvement  $EI_l(q)$ .

The two acquisition functions yield two GP strategies, including EI-based GP strategy (GP+EI) and UCB-based GP strategy (GP+UCB), which are shown in Algorithm 1. We first initialize GP by some starting points. In this work, we initialize the first two starting points by using GBS, *i.e.*,  $\{(q_1, y_1), (q_2, y_2)\}$  is obtained by using GBS. One can also use other techniques (*e.g.*, random) to obtain starting points. After the initialization, we update  $\mu_l(q)$  and  $\sigma_l^2(q)$  according to Equation 19 and Equation 20 based on the set of observed relevance feedback. Then we utilize the acquisition function to select a slot  $q_l$  according to Equation 21 or Equation 22. After that, we ask the question about  $q_l$  and observe the user relevance feedback  $y_l$ , and then add  $(q_l, y_l)$  to the set of observed relevance feedback to update  $\mu_l(q)$  and  $\sigma_l^2(q)$  again.

### 3.5 Item Ranking with ConvPS

After we train the embeddings of users, items, words, and slot-value pairs, we can retrieve items based on that. When a user  $u$  issues an initial query  $Q$ , our system first ranks an item  $v$  based on  $P(v|u, Q)$  according to Equation 12, in the initial iteration. After  $l$ -th round clarifying questions are selected to ask to the user, which is demonstrated as  $\mathcal{S}_{qa} = \{(q_1, a_1), (q_2, a_2), \dots, (q_l, a_l)\}$ , we rank an item  $v$  in the  $l$ -th iteration ( $l > 0$ ) according to:

$$P(v|u, Q, \mathcal{S}_{qa}) = \frac{\exp(v \cdot (\lambda_u \mathbf{u} + \lambda_Q Q + \lambda_c \sum_{(q,a) \in \mathcal{S}_{qa}} \mathbf{c}_{qa}))}{\sum_{v' \in \mathcal{V}} \exp(v' \cdot (\lambda_u \mathbf{u} + \lambda_Q Q + \lambda_c \sum_{(q,a) \in \mathcal{S}_{qa}} \mathbf{c}_{qa}))}. \quad (24)$$

## 4 EXPERIMENTS AND ANALYSIS

In this section, we start by introducing our experimental setup, including the datasets, evaluation metrics, baselines, conversation construction, parameter settings, and our research questions. We then demonstrate and analyze the results in terms of each of our research questions.

### 4.1 Experimental Setup

**4.1.1 Datasets.** For our experiments, we use the Amazon product datasets<sup>2</sup> [45], as in previous product search publications [3, 9, 63, 78]. The datasets contain millions of users and items. Each item contains rich metadata such as item descriptions, item categories, and user reviews. In this paper, we use three item categories with diverse scales from the Amazon product datasets, which are “Movies & TV”, “Health & Personal Care”, and “Cell Phones & Accessories”, following Bi et al. [9]. The former one constitutes a very large dataset while the rest are smaller, which allows us to test the effectiveness of our ConvPS model on collections of different sizes. The training/test set splitting for the three corresponding item categories is also the same with Bi et al. [9]. As there is no publicly available product search dataset containing search queries, we construct initial queries following the same paradigm employed in Ai et al. [3], Bi et al. [9], Van Gysel et al. [63], Zhang et al. [78], and Zhang et al. [78], *i.e.*, we use a subcategory title from the above three item categories to form a topic string as a query. Each subcategory (*i.e.*, query) includes multiple (relevant) items and each item can be relevant to multiple queries (subcategories). We evaluate the model performance for each user-query pair, *i.e.*, each individual user purchases belonging to the initial sub-category (query) observed in the data. If an item purchased by a user belongs to multiple sub-categories (queries), the pairing of this user with any of these queries creates valid user-query pairs. For each user-query pair, we rank all possible items within each individual category dataset. Therefore, the

<sup>2</sup><http://jmcauley.ucsd.edu/data/amazon/>

Table 3. Basic Statistics of our experimental datasets. Tr and Te denote the train and test sets, respectively. Arithmetic mean and standard deviation are indicated wherever applicable.

	<b>Cell Phones &amp; Accessories</b>	<b>Health &amp; Personal Care</b>	<b>Movies &amp; TV</b>
# Users	27,879	38,609	123,960
# Items	10,429	18,534	50,052
# Reviews	194,439	346,355	1,697,524
# Queries	165	779	248
# Words per query	$5.93 \pm 1.57$	$8.25 \pm 2.16$	$5.31 \pm 1.61$
# Slots (questions)	738	1,906	6,694
# Slot-value pairs	7,849	17,203	88,754
# User-query pairs	114,177 (Tr) 665 (Te)	231,186 (Tr) 282 (Te)	241,436 (Tr) 5,209 (Te)
# Relevant items per pair	$1.52 \pm 1.13$ (Tr) $1.00 \pm 0.05$ (Te)	$1.14 \pm 0.48$ (Tr) $1.00 \pm 0.00$ (Te)	$5.40 \pm 18.39$ (Tr) $1.10 \pm 0.49$ (Te)

number of candidate items is very large and target items for each user-query pair are very sparse, which leads to a difficult product search setting by nature. The statistics for our datasets are shown in Table 3.

**4.1.2 Evaluation Metrics.** As in Ai et al. [3], Bi et al. [9], and Zhang et al. [78], we use mean average precision (MAP), mean reciprocal rank (MRR) and normalized discounted cumulative gain (NDCG) as our evaluation metrics, where MAP and MRR are calculated based on top 100 items retrieved by each model, and NDCG is calculated based on top 10 items. Intuitively, MRR indicates the expected number of items that a user needs to browse before finding the first relevant item to purchase. MAP measures the overall performance taking both precision and recall into account. NDCG measures the ranking performance of results at the top of the ranking compared to an optimal ranked list. We are expecting the three measures to be strongly correlated.

**4.1.3 Baselines.** The following baselines are used:

- **LSE** [63], which is the first latent space model proposed for product search;
- **HEM** [3], which is a state-of-the-art retrieval model for personalized product search;
- **PMMN** [78], which is a state-of-the-art conversational search system asking questions on slot-value pairs.
- **AVLEM** [9], which is a state-of-the-art conversational product search model by incorporating negative feedback.

The first two baselines are two typical static baselines, which are widely used [9, 78]. The last two baselines are two state-of-the-art conversational baselines. We only include neural models on the basis of learning representations for soft-matching as our baselines since exact term matching has been proven insufficient to retrieve ideal products and term-based retrieval models are less effective in previous studies on product search [3, 9].

**4.1.4 Conversation Construction.** With the absence of the conversational product search dataset, we simulate users and construct conversations based on the Amazon product dataset, as done in Bi et al. [9], Zhang et al. [78], and Zou and Kanoulas [85]. We also conduct a small user study by involving real users described in Section 4.1.5. Specifically, we simulate conversations by adopting the approach of Bi et al. [9] and Zhang et al. [78], which extracts slot-value pairs from reviews of

items by using aspect-value pair extraction toolkit [79]. During training, all relevant (positive) slots for a user-item pair  $(u, v)$  are used to construct the questions. In order to train representations of slots with negative feedback, we also randomly select non-relevant (negative) slots from the entire slot pool for a user-item pair  $(u, v)$  to construct the questions. During testing time, we select the slots to form corresponding questions from the entire slot pool constructed by the training set, by using our proposed question selection strategies. We will never ask the slots that do not appear in the training set. For the answer during training and testing time, we simulate user feedback following previous literature [9, 59, 78]. They construct users' responses to the asked questions according to users' target items, which show their hidden intents. The system asks users their preferred value of a slot and answers are constructed according to the corresponding value for that slot in their target items. Therefore, same with them [9, 59, 78], we assume that a user has target items in mind. For an asked question on the basis of a slot, the users will respond with the corresponding value for that slot in their target items when the slot is contained in the target items, while responding with negative feedback "not relevant" when a slot is absent from the target items. In the case that the simulated user answers in the testing set are not in the training set, therefore leading to an embedding mismatching, we regard it as an invalid question and ignore it for updating the ranking results. Unlike [9, 78] only ask slots appearing in the target items, we ask slots from the entire slot pool (slots appearing in the entire item set), what we believe are not exclusive and more reasonable, as well as more challenging by nature.

**4.1.5 Online User Study.** Besides the simulated users, we develop an online Web system to conduct a small user study by involving real users. The purpose of the online user study is to validate some of assumptions in our conversational product search system and test the performance of our system "in-situ". Ideally, the existing customers of an online shopping platform who have previously purchased a number of products and now interact with our conversational product search system embedded in the shopping platform to locate their next target products would be optimal. In the absence of such a commercial product search system and user base, we utilize a crowdsourcing platform<sup>3</sup> to recruit crowd workers to interact with our online Web system. We first ask the crowd worker to choose a product category he/she feels comfortable with from our dataset. Then, we randomly sample a product from our test data in the selected product category as a target product. The user is then guided to carefully review the information about the target product, e.g., a product image, title, description, and the extracted slot-value pairs. Once the user indicates that he/she is familiar with the target product, the target product disappears from the screen and the conversation with the system starts. Then, a question selected by our algorithm is asked to the user. In this user study, we use our ConvPS algorithm with LinRel question selection strategy. After the user observes the algorithmically chosen question, he/she can provide an answer to the question according to the information of the target product. The user has the option to stop answering questions at any time during her interaction with the system. When the interaction with the system stops, we ask users a series of exit questions regarding their experiences.

**4.1.6 Parameter Settings.** We set the hyper-parameters to the combination that achieved the highest pairwise accuracy in the offline observations as follows: we train our ConvPS model with stochastic gradient descent for 20 epochs with batch size 64. For stochastic gradient descent in training, the initial learning rate is set as 0.5 and gradually decreases to 0. We use the global norm clip of gradients with 5 for stable training. We set hyper-parameters  $\lambda_I = e_i = 0.1$ , and  $\lambda_u = \lambda_Q = \lambda_c = 1$ . The number of negative samples in Equation 14 is set as 5 and L2 regularization strength  $\gamma$  is tuned from 0 to 0.01. As in Ai et al. [3], and Bi et al. [9], we set the subsampling rate of words for "Movies

<sup>3</sup><https://www.prolific.co/>

Table 4. Performance comparison between our ConvPS model and baselines (LSE, HEM, PMMN, and AVLEM) on the “Cell Phones & Accessories”, “Health & Personal Care”, and “Movies & TV” categories. The performance of PMMN, AVLEM and our ConvPS model are reported when 5 questions (slots) are asked. ‘\*’ indicates significant differences between ConvNF and ConvPS<sub>init</sub> in Fisher random test with  $p$ -value  $< 0.05$ , ‘†’ denotes significant improvements upon the best baseline with  $p$ -value  $< 0.05$ . Best performances are in bold.

Model	Cell Phones & Accessories			Health & Personal Care			Movies & TV		
	MAP	MRR	NDCG	MAP	MRR	NDCG	MAP	MRR	NDCG
LSE	0.098	0.098	0.084	0.155	0.157	0.195	0.023	0.025	0.027
HEM	0.115	0.115	0.116	0.189	0.189	0.201	0.026	0.030	0.030
PMMN	0.115	0.116	0.122	0.180	0.180	0.196	0.029	0.030	0.023
AVLEM	0.125	0.125	0.130	0.214	0.214	0.251	0.025	0.026	0.027
ConvPS <sub>init</sub>	0.126	0.126	0.136	0.243	0.243	0.267	0.021	0.023	0.024
ConvPS <sub>random</sub>	0.131*†	0.131*†	0.141*†	0.254*†	0.254*†	0.279*†	0.022	0.023	0.024
ConvPS <sub>GBS</sub>	0.226*†	0.226*†	0.258*†	0.340*†	0.340*†	0.359*†	0.042*†	0.045*†	0.048*†
ConvPS <sub>LinRel</sub>	<b>0.235*†</b>	<b>0.236*†</b>	<b>0.265*†</b>	0.401*†	0.401*†	0.413*†	<b>0.044*†</b>	<b>0.046*†</b>	<b>0.050*†</b>
ConvPS <sub>GP+EI</sub>	0.223*†	0.223*†	0.242*†	<b>0.408*†</b>	<b>0.408*†</b>	<b>0.441*†</b>	0.035*†	0.037*†	0.039*†
ConvPS <sub>GP+UCB</sub>	0.231*†	0.231*†	0.251*†	0.347*†	0.347*†	0.376*†	0.035*†	0.038*†	0.040*†

& TV”, and “Cell Phones & Accessories” as  $10^{-6}$  and  $10^{-5}$ , respectively, to speed up the training and reduce the effect of common words. We study the impact of the number of questions asked, the embedding size, the batch size, and the explore-exploit trade-off parameter  $c$  in Section 4.5. The number of questions asked for the conversation is examined from 0 to 10 with step size 1, the embedding size is examined from 100 to 500 with step size 100, the batch size is tuned from [64, 128, 256, 612], the explore-exploit trade-off parameters  $c$  and  $\beta$  are tested from [0, 2, 4, 6, 8, 10]. Unless mentioned otherwise, we use 5 questions and other optimal parameters, which are 200 for the embedding size, 64 for the batch size, and 4, 2 for the explore-exploit trade-off parameter  $c$  and  $\beta$  respectively, to report results. For the parameter settings of the four baselines, we use the optimal parameters stated in the corresponding product search publications and tuned their hyper-parameters in the same way as they reported.

**4.1.7 Research Questions.** In our experiments, we will investigate and answer the following four questions:

**RQ1** How effective is our proposed model ConvPS compared to prior works?

**RQ2** Which question selection strategy is the best?

**RQ3** What are the effects of different model components?

**RQ4** How do the parameters of ConvPS affect its efficacy?

## 4.2 RQ1 Overall Performance Comparison

To study the overall effectiveness of our ConvPS model, we compare the performance of our ConvPS, measured by MAP, MRR, and NDCG, with baselines shown in Section 4.1.3. The results on the “Cell Phones & Accessories”, “Health & Personal Care”, and “Movies & TV” categories are shown in Table 4. We refer to the ConvPS with random question selection, with GBS question selection strategy, with LinRel question selection strategy, with EI-based GP question selection strategy, and with UCB-based GP question selection strategy as ConvPS<sub>random</sub>, ConvPS<sub>GBS</sub>, ConvPS<sub>LinRel</sub>, ConvPS<sub>GP+EI</sub>, and ConvPS<sub>GP+UCB</sub>, respectively.

As shown in Table 4, we observe that our ConvPS model with different question selection strategies (ConvPS<sub>random</sub>, ConvPS<sub>GBS</sub>, ConvPS<sub>LinRel</sub>, ConvPS<sub>GP+EI</sub>, and ConvPS<sub>GP+UCB</sub>) outperforms the baselines LSE, HEM, PMMN, and AVLEM, on all three metrics, which suggests that our ConvPS

Table 5. Question selection strategy comparison in terms of the ratio of questions with positive feedback, negative feedback, or invalid feedback.

Model	% Positive	% Negative	% Invalid
ConvPS <sub>random</sub>	5.0%	0.9%	94.1%
ConvPS <sub>GBS</sub>	69.6%	15.1%	15.3%
ConvPS <sub>LinRel</sub>	71.0%	26.0%	3.0%
ConvPS <sub>GP+EI</sub>	64.1%	11.0%	24.9%
ConvPS <sub>GP+UCB</sub>	64.0%	11.1%	24.9%

model is effective. As expected, with 5 questions asked, our ConvPS model with different question selection strategies significantly outperforms the static baselines LSE and HEM. We attribute this to the reason that our ConvPS can continuously learn from the user and clear users' item needs through conversations. With 0 questions asked, our ConvPS<sub>init</sub> model can still achieve comparable performance on "Movies & TV" and better performance on "Cell Phones & Accessories" and "Health & Personal Care". Further, our ConvPS models perform better than the two dynamic baselines PMMN and AVLEM. This suggests that our ConvPS models better learn user true preferences. This might be explained by the fact that the embeddings (*i.e.*, user embeddings, query embeddings, item embeddings, and conversation embeddings in terms of slot-value embeddings) learned by our ConvPS models can better distinguish the target items from those irrelevant items, or our question selection strategies can better gather informative user feedback. Compared with ConvPS<sub>init</sub>, we observe consistent and large improvements of our ConvPS model with the four different question selection strategies (*i.e.*, ConvPS<sub>GBS</sub>, ConvPS<sub>LinRel</sub>, ConvPS<sub>GP+EI</sub>, and ConvPS<sub>GP+UCB</sub>), which is approximately 100% in terms of three metrics in all three categories. This clearly indicates the contribution of conversations, suggesting conversational product search is highly beneficial.

### 4.3 RQ2 Question Selection Strategy Comparison

In Figure 2, we report the comparison results of the various question selection strategies on "Cell Phones & Accessories". We see that our ConvPS models outperform PMMN and AVLEM and the performance gains become larger as the number of questions increases. What we indeed observed is that the performance on all three metrics regarding PMMN and AVLEM increases slowly while the performance of our ConvPS models improves greatly, as the number of questions increases. This again demonstrates that our ConvPS models learn more effectively and efficiently from the user than PMMN and AVLEM, reflected in either representation learning or question selection strategies.

ConvPS<sub>GBS</sub>, ConvPS<sub>LinRel</sub>, ConvPS<sub>GP+EI</sub>, and ConvPS<sub>GP+UCB</sub> achieve much higher performance than ConvPS<sub>random</sub>, demonstrating our four question selection strategies are highly beneficial. ConvPS<sub>random</sub> learns slowly, because it fails to select more promising questions to ask for effective learning from the user. Overall, ConvPS<sub>GBS</sub>, ConvPS<sub>LinRel</sub>, ConvPS<sub>GP+EI</sub>, and ConvPS<sub>GP+UCB</sub> perform well and achieve comparable performance. Specifically, we find that the bandit-inspired strategies ConvPS<sub>LinRel</sub> and ConvPS<sub>GP+UCB</sub> perform slightly better at early stage (*e.g.*, less than 6 questions). The plausible reason is that ConvPS<sub>LinRel</sub> and ConvPS<sub>GP+UCB</sub> systematically balance the need to explore new solutions with the need to reap the rewards of what has already been learned. At latter stage (*e.g.*, more than 6 questions), ConvPS<sub>GBS</sub> performs slightly better. This might be because the GBS question selection strategy can learn from user preference over items  $\pi_l(v)$  as the number of questions increases. With the user preference more close to the true user preference at the latter stage, GBS learns faster than other question selection strategies.

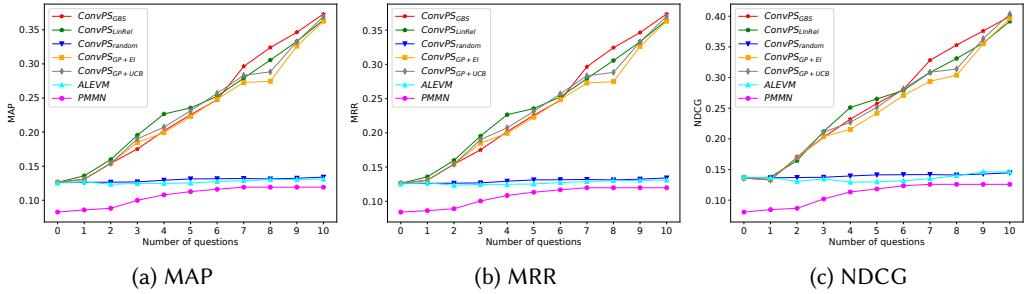


Fig. 2. Performance of different question selection strategies.

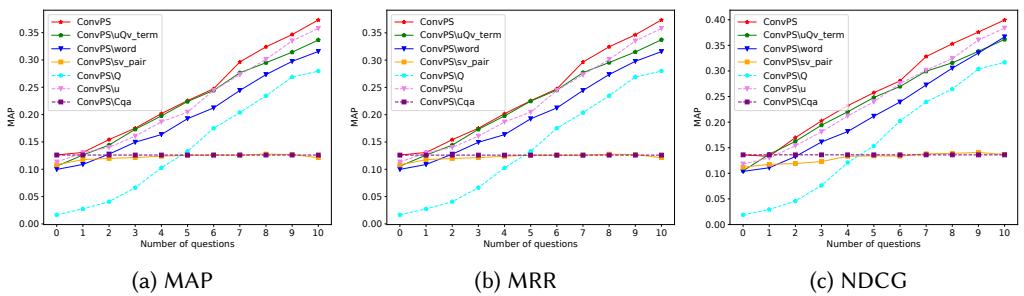


Fig. 3. The performance of ConvPS with different components removed.

We further explore why those question selection strategies perform differently by reporting the ratio of questions with positive feedback, negative feedback, or invalid feedback. Ideally, the user is more engaged to be asked those questions with positive feedback, while less engaged be asked questions with negative or invalid feedback. Questions with negative feedback usually mean those questions are irrelevant. Due to the great diversity in questions and items, most questions are irrelevant to a target item and they should not be asked to dissatisfy users. Questions with invalid feedback mean those questions are useless leading to no gain. As shown in Table 5, we observe that ConvPS<sub>LinRel</sub> achieve top performance across models, with the highest ratio of questions with positive feedback, and the lowest ratio of questions with invalid feedback. ConvPS<sub>random</sub> asks a large number of questions with invalid feedback, leading to the lowest performance in Figure 2. This is because the question pool is large and the percentage of questions with positive feedback in the question pool for a certain target item is low.

#### 4.4 RQ3 Ablation Study

To evaluate the importance of different components of our ConvPS, we conduct a series of ablation experiments. We refer the ConvPS by removing the user and item language models for textual descriptions (*i.e.*, removing the corresponding terms of Equation 3 and Equation 4 from Equation 13), by removing the user and item language models for slot-value pairs (*i.e.*, removing the corresponding terms of Equation 5 and Equation 6 from Equation 13), and by removing the non-conversational term (*i.e.*, removing Equation 12 from Equation 13), as ConvPS\word, ConvPS\sv\_pair, and ConvPS\uQv\_term, respectively. The performances of ConvPS with different components removed are shown in Figure 3. Due to the similar trends in the three categories, we

illustrate the trend on the “Cell Phones & Accessories” category only and omit the other two categories. For the results reported, we use GBS for the question selection. From what we observed in Figure 3, removing the user and item language models for textual descriptions (*i.e.*, ConvPS\word), removing the user and item language models for slot-value pairs (*i.e.*, ConvPS\sv\_pair), or removing the non-conversational term (*i.e.*, ConvPS\uQv\_term) leads to inferior retrieval performance both before feedback (*i.e.*, the number of questions is 0) and after feedback(*i.e.*, the number of questions is from 1 to 10). This indicates that these three components are crucial. We also see that ConvPS\sv\_pair contributes to the most significant improvement to the final performance. Removing the user and item language models for slot-value pairs leads to the highest decrease for the performance, and its performance does not increase much as the number of questions increases. This might be because removing the user and item language models for slot-value pairs greatly affects the representation learning of slot-value pairs, and thus longer conversations (more number of questions) do not obtain better performance gains. This indicates that poor representation learning introduces a risk although good representation learning of conversations leads to a benefit for the search performance [70, 83, 87].

Furthermore, we remove the learned representations of users, queries, slot-value pairs to perform item ranking with ConvPS during testing. We refer the ConvPS by removing the learned user representations (*i.e.*, removing the user representations from Equation 24), by removing the learned query representations (*i.e.*, removing the query representations from Equation 24), and by removing the learned slot-value pair representations (*i.e.*, removing the slot-value pair representations from Equation 24), as ConvPS\mathbf{u}, ConvPS\mathbf{Q}, and ConvPS\mathbf{C}\_{qa}, respectively. As shown in Figure 3, we observe that removing any of the learned representations of users, queries, or slot-value pairs leads to worse performance to retrieve items, especially when removing the learned query representations and the learned slot-value pair representations. This indicates that our learned representations are effective.

## 4.5 RQ4 Parameter Sensitivity

For parameter sensitivity, we explore the effects of main parameters in our work, including (1) the impact of the number of questions, (2) the impact of embedding size, (3) the impact of batch size, and (4) the impact of explore-exploit trade-off. Due to the similar trends in the three categories, we illustrate the trend on the “Cell Phones & Accessories” category only and omit the other two categories.

**4.5.1 Impact of the Number of Questions.** From Figure 2, we observe that the performance of ConvPS greatly improves as the number of questions increases. This might be because that higher number of questions gathers more information from users, leading to more close to users’ actual needs.

**4.5.2 Impact of Embedding Size.** Figure 4 shows the sensitivity of our ConvPS model in terms of embedding size. For the results reported, we use GBS for the question selection with 5 questions asked. As we observed, with the embedding size increasing, the performance of our ConvPS model, measured by the three metrics, first increases and then decreases. The ConvPS model achieves the best performance when the embedding size is equal to 200. Moreover, the performance gains obtained from the feedback process for our model (ConvPS vs. ConvPS<sub>init</sub>) are stable with respect to the changes of embedding sizes.

**4.5.3 Impact of Batch Size.** Figure 5 shows the sensitivity of our ConvPS model in terms of batch size. Again, For the results reported, we use GBS for the question selection with 5 questions asked. We observe a decrease in the model performance on the three metrics after increasing the batch

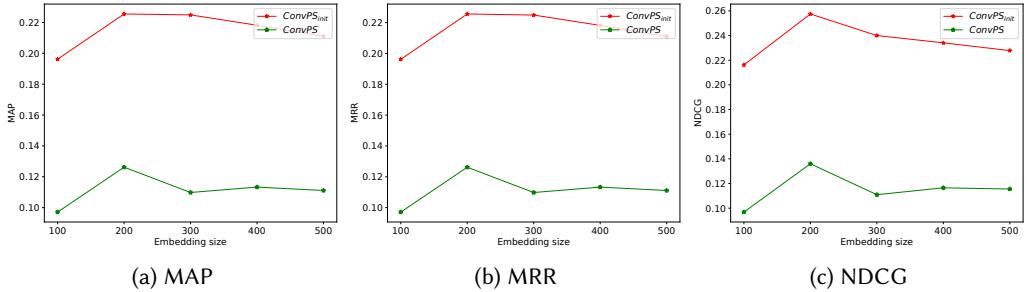


Fig. 4. Effect of embedding sizes.

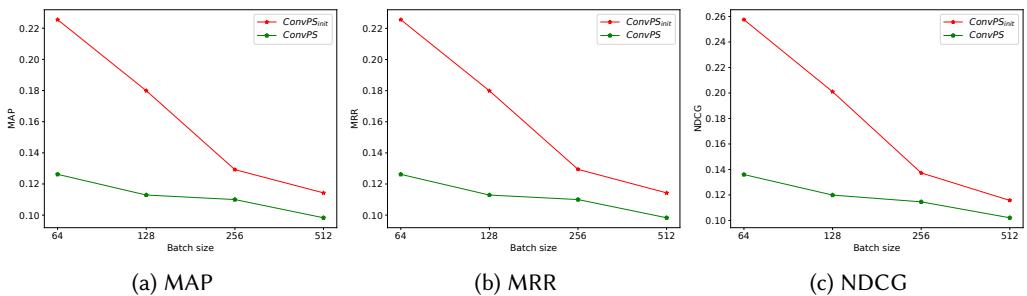


Fig. 5. Effect of batch sizes.

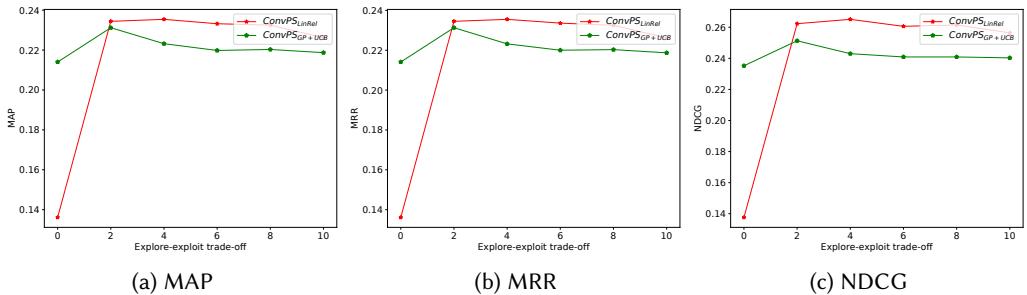


Fig. 6. Effect of explore-exploit trade-off.

size from 64 to 512. The ConvPS model after 5 questions asked always leads to higher performance compared with ConvPS<sub>init</sub>, and the performance gap between them becomes smaller in the case of a bigger batch size.

**4.5.4 Impact of Explore-exploit Trade-off.** We explore the effect of explore-exploit trade-off in ConvPS<sub>LinRel</sub> and ConvPS<sub>GP+UCB</sub>, as shown in Figure 6. We observe that the performance first increases and then decreases for both ConvPS<sub>LinRel</sub> and ConvPS<sub>GP+UCB</sub>. The optimal explore-exploit trade-offs are 4 and 2 for ConvPS<sub>LinRel</sub> and ConvPS<sub>GP+UCB</sub>, respectively. When the explore-exploit trade-off is set to zero, which means no exploration, both models achieve the lowest performance. This might be because bandit strategies like LinRel and UCB tend to ask similar questions with previous ones when there is no exploration, preventing truly effective learning from conversations,

Table 6. System effectiveness on user study.

# of questions	MAP	MRR	NDCG
5	0.249	0.250	0.283
stopping	0.375	0.376	0.411

which results in a lower performance gain. This highlights the importance of exploration for learning to ask questions. Compared with ConvPS<sub>GP+UCB</sub>, ConvPS<sub>LinRel</sub> is more sensitive in the case of no exploration (*i.e.*, the explore-exploit trade-off is set to zero) while is less sensitive with the explore-exploit trade-off parameter when exploration is included (*i.e.*, the explore-exploit trade-off is larger than zero).

#### 4.6 Online User Study

By the online user study, we first examine whether real users are willing to answer the system-generated questions and how well they can answer these questions. We collected 552 conversations made between our system and 26 crowd workers<sup>4</sup> on 59 target items. From the collected data, we see that users answered an average number of 9 questions and a median number of 10 questions per target item during the interaction with the system. In addition, in the exit questionnaire, the number of questions that users indicated they are willing to answer is 6 on average. Most of the users (64.3%) declare that they are willing to answer at least 5 questions for locating their target products. For how well users can answer the system-generated questions, we observe that most (71.4%) of the users indicate that the system’s questions were easy to answer in the exit questionnaire. In most cases, users provided the correct answers to the system’s questions (77%). They were not sure about their answers 11% of the time, and they provide the wrong answers (*i.e.*, their answers disagreed with the product information) 12% of the time.

Secondly, we evaluate the system performance, measured by the three evaluation metrics, using the collected user study data. The performance with 5 questions asked, as well as the performance when the user stopped answering questions are shown in Table 6. From the results, we observe that the performance is in agreement with the offline evaluation performance of ConvPS<sub>LinRel</sub> in Table 4.

### 5 CONCLUSIONS AND DISCUSSIONS

In this paper, we propose a new conversational product search model, ConvPS. The ConvPS model firstly learns the representations of users, queries, items, and conversations (in terms of slot-value pairs) via a unified generative framework, and then utilizes the trained representations to retrieve the target items in the latent semantic space. Four learning to ask strategies are also incorporated into the ConvPS model to select a sequence of effective questions to ask to the user. Experimental results on the Amazon product datasets demonstrate that the conversation in product search is beneficial and there is a significant improvement of the ConvPS model compared with existing baselines.

One limitation of this work is that we used the simulated data like previous work [9, 78, 85], because of the difficulty of obtaining such data in a real scenario. However, as early exploration in conversational product search, we demonstrate that conversational product search by incorporating user feedback on slot-value pairs via representation learning is effective and it is a promising research direction.

<sup>4</sup>Their demographic data varied as follows: gender: 20 females, and 6 males; age: 7 participants were 18–23 years old, 3 were 24–27, 7 were 28–35, and 9 were older than 35; English language proficiency: 20 native and 6 proficient.

In this work, we construct questions by using templates on the basis of slot-value pairs. The template-based approach is simple yet effective [74]. However, one can also use other natural language question generation techniques to construct questions, e.g., generating diverse and specific clarifying questions with keywords based on product description [81]. We ask questions over slots and collect user feedback over corresponding values, as in past work in the field [59, 78]. We make the assumption that the user’s feedback is positive or negative, which may not always be the case, e.g., in the case the user is not certain about the answer to a question; hence, it is worth extending the system incorporating user responses other than slot-value pairs and free-form written feedback in the future work. Also, users may provide some custom values (along with/without the system provided values) which are out of the training vocabulary leading to an embedding mismatch. In this work we regard them as invalid questions and ignore their answers. One can also use a unified embedding or pre-trained embedding (e.g., from BERT [18]) or use nearby embeddings to generalize the embedding of the mismatched value, which we leave for future work. Moreover, in this work we do not focus on natural language understanding, as done in Bi et al. [9], Zhang et al. [78]. We leave the natural language understanding techniques to translate user response to slot-value pairs as future work. Although slot-value pairs are effective and typically used in conversational systems (e.g., slot-value pairs are usually used for dialogue state tracking in task-oriented dialogue systems) [59, 82], the natural language understanding from user response to slot-value pairs may not be 100% accurate. It is thus worth extending the system by incorporating and modeling uncertainty and noise [86].

The question selection strategies are independent (or indirectly correlated) from the trained embeddings, to maintain wide adaptability. However, one can also incorporate the trained embeddings to the objective functions of question selection strategies, so that taking the trained embeddings into account when selecting questions to ask, or simply using the trained embeddings to filter out those questions with similar embeddings based on the assumption that asking another question with similar embedding is not able to add new information to locate the target items.

A lot of research has been done on respecting ethical implications and preserving privacy. Since our conversational product search model is personalized and based on word/slot-value embeddings, it may introduce personalization bias and embedding bias [22], leading to the system favoring some items over others. It is thus worth incorporating debiasing techniques e.g., debiasing embeddings [11], controlling personalization [2], and diversifying the response [22] to mitigate this issue. Also, when collecting user data by asking CQs, the ethical and privacy violations may occur. A typical solution is to anonymize the data and try to hide the identity of users [13], and encourage users to be careful about providing answers for ethical-sensitive and privacy-sensitive CQs. A privacy preserving model can also be trained when adding noise intentionally to the supervision signal to protect user privacy [21].

## ACKNOWLEDGEMENT

We gratefully appreciate the associate editor and three anonymous reviewers for their valuable and detailed comments that helped to improve the quality of this article. This work was partly done when the first author was a visiting Ph.D. student at the Information Retrieval and Knowledge Management Research Lab, York University, Canada. This research was supported by the NWO Smart Culture - Big Data / Digital Humanities (314-99-301), the NWO Innovational Research Incentives Scheme Vidi (016.Vidi.189.039), the H2020-EU.3.4. - SOCIETAL CHALLENGES - Smart, Green And Integrated Transport (814961), the Natural Science Foundation of China (62402093, 61902219), and the Google Faculty Research Awards program. This research was also supported by the research grant (RGPIN-2020-07157) from Natural Sciences and Engineering Research Council (NSERC) and York Research Chairs (YRC) program. All contents represent the opinion of the

authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors. We also acknowledge Compute Canada for providing us with the computing resources to conduct experiments.

## REFERENCES

- [1] Aman Ahuja, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K Reddy. 2020. Language-Agnostic Representation Learning for Product Search on E-Commerce Platforms. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 7–15.
- [2] Qingyao Ai, Daniel N Hill, SVN Vishwanathan, and W Bruce Croft. 2019. A zero attention model for personalized product search. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 379–388.
- [3] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and W Bruce Croft. 2017. Learning a hierarchical embedding model for personalized product search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 645–654.
- [4] Qingyao Ai, Yongfeng Zhang, Keping Bi, and W Bruce Croft. 2019. Explainable product search with a dynamic relation embedding model. *ACM Transactions on Information Systems (TOIS)* 38, 1 (2019), 1–29.
- [5] Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. 2020. ConvAI3: Generating Clarifying Questions for Open-Domain Dialogue Systems (ClariQ). *arXiv preprint arXiv:2009.11352* (2020).
- [6] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W Bruce Croft. 2019. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 475–484.
- [7] Peter Auer. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3, Nov (2002), 397–422.
- [8] N. Belkin, Colleen Cool, A. Stein, and U. Thiel. 1995. Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. *Expert Systems With Applications* 9 (1995), 379–395.
- [9] Keping Bi, Qingyao Ai, Yongfeng Zhang, and W Bruce Croft. 2019. Conversational product search based on negative feedback. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 359–368.
- [10] Keping Bi, Choon Hui Teo, Yesh Dattatreya, Vijai Mohan, and W Bruce Croft. 2019. A Study of Context Dependencies in Multi-page Product Search. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2333–2336.
- [11] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems* 29 (2016).
- [12] David Carmel, Elad Haramaty, Arnon Lazerson, Liane Lewin-Eytan, and Yoelle Maarek. 2020. Why do people buy seemingly irrelevant items in voice product search? On the relation between product relevance and customer satisfaction in ecommerce. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 79–87.
- [13] Claudio Carpineto and Giovanni Romano. 2013. Semantic search log k-anonymization with generalized k-cores of query concept graph. In *European Conference on Information Retrieval*. 110–121.
- [14] Yihong Chen, Bei Chen, Xuguang Duan, Jian-Guang Lou, Yue Wang, Wenwu Zhu, and Yong Cao. 2018. Learning-to-Ask: Knowledge Acquisition via 20 Questions. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) (KDD '18). 1216–1225.
- [15] Konstantina Christakopoulou and Arindam Banerjee. 2018. Learning to interact with users: A collaborative-bandit approach. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 612–620.
- [16] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 815–824.
- [17] Jeffrey Dalton, Chenyan Xiong, Vaibhav Kumar, and Jamie Callan. 2020. Cast-19: A dataset for conversational information seeking. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1985–1988.
- [18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
- [19] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. A probabilistic mixture model for mining and analyzing product search log. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (San Francisco, California, USA) (CIKM '13). 2179–2188.
- [20] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. Supporting Keyword Search in Product Database: A Probabilistic Approach. *Proc. VLDB Endow.* 6, 14 (Sept. 2013), 1786–1797.

- [21] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2014. Privacy aware learning. *Journal of the ACM (JACM)* 61, 6 (2014), 1–57.
- [22] Emma J Gerritsse, Faegheh Hasibi, and Arjen P de Vries. 2020. Bias in conversational search: The double-edged sword of the personalized knowledge graph. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*. 133–136.
- [23] Dorota Glowacka, Tuukka Ruotsalo, Ksenia Konuyshkova, Kumaripaba Athukorala, Samuel Kaski, and Giulio Jacucci. 2013. Directing exploratory search: Reinforcement learning from user interactions with keywords. In *Proceedings of the 2013 international conference on Intelligent user interfaces*. 117–128.
- [24] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Yinglong Wang, Jun Ma, and Mohan Kankanhalli. 2019. Attentive Long Short-Term Preference Modeling for Personalized Product Search. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019), 19.
- [25] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Xin-Shun Xu, and Mohan Kankanhalli. 2018. Multi-modal Preference Modeling for Product Search. In *Proceedings of the 26th ACM International Conference on Multimedia* (Seoul, Republic of Korea) (MM ’18). 1865–1873.
- [26] Helia Hashemi, Hamed Zamani, and W. Bruce Croft. 2020. Guided Transformer: Leveraging Multiple External Sources for Representation Learning in Conversational Search. In *SIGIR*. 1131–1140.
- [27] Ben He, Jimmy Xiangji Huang, and Xiaofeng Zhou. 2011. Modeling term proximity for probabilistic information retrieval models. *Inf. Sci.* 181, 14 (2011), 3017–3031.
- [28] Huang Hu, Xianchao Wu, Bingfeng Luo, Chongyang Tao, Can Xu, Wei Wu, and Zhan Chen. 2018. Playing 20 Question Game with Policy-Based Reinforcement Learning. *arXiv preprint arXiv:1808.07645* (2018).
- [29] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 368–377.
- [30] Xiangji Huang, Fuchun Peng, Dale Schuurmans, Nick Cercone, and Stephen E. Robertson. 2003. Applying Machine Learning to Text Segmentation for Information Retrieval. *Inf. Retr.* 6, 3-4 (2003), 333–362.
- [31] Andrea Iovine, Pasquale Lops, Fedelucio Narducci, Marco de Gemmis, and Giovanni Semeraro. 2021. An empirical evaluation of active learning strategies for profile elicitation in a conversational recommender system. *Journal of Intelligent Information Systems* (2021), 1–26.
- [32] Ruwang Jiao, Sanyou Zeng, Changhe Li, Yuhong Jiang, and Yaochu Jin. 2019. A complete expected improvement criterion for Gaussian process assisted highly constrained expensive optimization. *Information Sciences* 471 (2019), 80–96.
- [33] Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and ChengXiang Zhai. 2017. On application of learning to rank for e-commerce search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 475–484.
- [34] Tom Kenter and Maarten de Rijke. 2017. Attentive memory networks: Efficient machine reading for conversational search. *arXiv preprint arXiv:1712.07229* (2017).
- [35] Kimiya Keyvan and Jimmy Xiangji Huang. 2022. How to Approach Ambiguous Queries in Conversational Search? A Survey of Techniques, Approaches, Tools and Challenges. *ACM Comput. Surv.* (may 2022).
- [36] Johannes Kiesel, Arefeh Bahrami, Benno Stein, Avishek Anand, and Matthias Hagen. 2018. Toward voice query clarification. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1257–1260.
- [37] Antonios Minas Krasakis, Mohammad Aliannejadi, Nikos Voskarides, and Evangelos Kanoulas. 2020. Analysing the effect of clarifying questions on document ranking in conversational search. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*. 129–132.
- [38] Hyun-Hwa Lee and Jihyun Kim. 2008. The effects of shopping orientations on consumers' satisfaction with product search and purchases in a multi-channel environment. *Journal of Fashion Marketing and Management: An International Journal* 12, 2 (2008), 193–216.
- [39] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems* 27 (2014), 2177–2185.
- [40] Dan Li, Zhaochun Ren, and Evangelos Kanoulas. 2021. CrowdGP: a Gaussian Process Model for Inferring Relevance from Crowd Annotations. In *Proceedings of the Web Conference 2021*. 1821–1832.
- [41] Hang Li and Jun Xu. 2014. Semantic matching in search. *Foundations and Trends in Information retrieval* 7, 5 (2014), 343–469.
- [42] Shijun Li, Wenqiang Lei, Qingyun Wu, Xiangnan He, Peng Jiang, and Tat-Seng Chua. 2021. Seamlessly unifying attributes and items: Conversational recommendation for cold-start users. *ACM Transactions on Information Systems (TOIS)* 39, 4 (2021), 1–29.

- [43] Soon Chong Johnson Lim, Ying Liu, and Wing Bun Lee. 2010. Multi-facet product information search and retrieval using semantically annotated product family ontology. *Information Processing & Management* 46, 4 (2010), 479 – 493.
- [44] Aldo Lipani, Ben Carterette, and Emine Yilmaz. 2021. How am I doing?: Evaluating conversational search systems offline. *ACM Transactions on Information Systems (TOIS)* 39, 4 (2021), 1–22.
- [45] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 43–52.
- [46] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems* 26 (2013), 3111–3119.
- [47] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.
- [48] Robert Nowak. 2008. Generalized binary search. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*. 568–574.
- [49] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24, 4 (2016), 694–707.
- [50] Jay M Ponte and W Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 275–281.
- [51] Chen Qu, Liu Yang, W Bruce Croft, Johanne R Trippas, Yongfeng Zhang, and Minghui Qiu. 2018. Analyzing and characterizing user intent in information-seeking conversations. In *The 41st international acm sigir conference on research & development in information retrieval*. 989–992.
- [52] Filip Radlinski and Nick Craswell. 2017. A theoretical framework for conversational search. In *Proceedings of the 2017 conference on conference human information interaction and retrieval*. 117–126.
- [53] Sudha Rao and Hal Daumé III. 2018. Learning to Ask Good Questions: Ranking Clarification Questions using Neural Expected Value of Perfect Information. *arXiv preprint arXiv:1805.04655* (2018).
- [54] Pengjie Ren, Zhongkun Liu, Xiaomeng Song, Hongtao Tian, Zhumin Chen, Zhaochun Ren, and Maarten de Rijke. 2021. Wizard of Search Engine: Access to Information Through Conversations with Search Engines. *arXiv preprint arXiv:2105.08301* (2021).
- [55] Xuhui Ren, Hongzhi Yin, Tong Chen, Hao Wang, Zi Huang, and Kai Zheng. 2021. Learning to Ask Appropriate Questions in Conversational Recommendation. *arXiv preprint arXiv:2105.04774* (2021).
- [56] Corbin Rosset, Chenyan Xiong, Xia Song, Daniel Campos, Nick Craswell, Saurabh Tiwary, and Paul Bennett. 2020. Leading Conversational Search by Suggesting Useful Questions. In *Proceedings of The Web Conference 2020* (Taipei, Taiwan) (*WWW '20*). 1160–1170.
- [57] Jennifer Rowley. 2000. Product search in e-shopping: a review and research propositions. *Journal of consumer marketing* (2000).
- [58] Tuukka Ruotsalo, Jaakko Peltonen, Manuel JA Eugster, Dorota Glowacka, Patrik Floréen, Petri Myllymäki, Giulio Jacucci, and Samuel Kaski. 2018. Interactive intent modeling for exploratory search. *ACM Transactions on Information Systems (TOIS)* 36, 4 (2018), 1–46.
- [59] Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *The 41st international acm sigir conference on research & development in information retrieval*. 235–244.
- [60] Shan Suthaharan. 2016. Support vector machine. In *Machine learning models and algorithms for big data classification*. Springer, 207–235.
- [61] Simon Tong. 2001. *Active learning: theory and applications*. Stanford University.
- [62] Johanne R. Trippas, Damiano Spina, Lawrence Cavedon, and Mark Sanderson. 2017. How Do People Interact in Conversational Speech-Only Search Tasks: A Preliminary Analysis. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval* (Oslo, Norway) (*CHIIR '17*). 325–328.
- [63] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning Latent Vector Spaces for Product Search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, Indiana, USA) (*CIKM '16*). 165–174.
- [64] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2018. Mix 'n Match: Integrating Text Matching and Product Substitutability within Product Search. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy) (*CIKM '18*). 1373–1382.
- [65] Damir Vandic, Flavius Frasincar, and Uzay Kaymak. 2013. Facet Selection Algorithms for Web Product Search. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management* (San Francisco, California, USA) (*CIKM '13*). 2327–2332.
- [66] Damir Vandic, Jan-Willem van Dam, and Flavius Frasincar. 2012. Faceted product search powered by the Semantic Web. *Decision Support Systems* 53, 3 (2012), 425 – 437.

- [67] Alexandra Vtyurina, Denis Savenkov, Eugene Agichtein, and Charles LA Clarke. 2017. Exploring conversational search with humans, assistants, and wizards. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2187–2193.
- [68] Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 363–372.
- [69] Qi Wang, Xinchen Liu, Wu Liu, An-An Liu, Wenyin Liu, and Tao Mei. 2020. MetaSearch: Incremental Product Search via Deep Meta-Learning. *IEEE Transactions on Image Processing* 29 (2020), 7549–7564.
- [70] Zhenduo Wang and Qingyao Ai. 2021. Controlling the Risk of Conversational Search via Reinforcement Learning. In *Proceedings of the Web Conference 2021*. 1968–1977.
- [71] Liang Wu, Diane Hu, Liangjie Hong, and Huan Liu. 2018. Turning clicks into purchases: Revenue optimization for product search in e-commerce. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 365–374.
- [72] Teng Xiao, Jiaxin Ren, Zaiqiao Meng, Huan Sun, and Shangsong Liang. 2019. Dynamic bayesian metric learning for personalized product search. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1693–1702.
- [73] Jun Yu, Sunil Mohan, Duangmanee Putthividhya, and Weng-Keen Wong. 2014. Latent dirichlet allocation based diversified retrieval for e-commerce search. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 463–472.
- [74] Hamed Zamani, Susan Dumais, Nick Craswell, Paul Bennett, and Gord Lueck. 2020. Generating clarifying questions for information retrieval. In *Proceedings of The Web Conference 2020*. 418–428.
- [75] Hamed Zamani, Gord Lueck, Everest Chen, Rodolfo Quispe, Flint Luu, and Nick Craswell. 2020. Mimics: A large-scale data collection for search clarification. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 3189–3196.
- [76] Hamed Zamani, Bhaskar Mitra, Everest Chen, Gord Lueck, Fernando Diaz, Paul N Bennett, Nick Craswell, and Susan T Dumais. 2020. Analyzing and Learning from User Interactions for Search Clarification. *arXiv preprint arXiv:2006.00166* (2020).
- [77] Xiaoying Zhang, Hong Xie, Hang Li, and John CS Lui. 2020. Conversational contextual bandit: Algorithm and application. In *Proceedings of The Web Conference 2020*. 662–672.
- [78] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W Bruce Croft. 2018. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 177–186.
- [79] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. 83–92.
- [80] Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural IR Meets Graph Embedding: A Ranking Model for Product Search. *arXiv preprint arXiv:1901.08286* (2019).
- [81] Zhiling Zhang and Kenny Q Zhu. 2021. Diverse and Specific Clarification Question Generation with Keywords. *arXiv preprint arXiv:2104.10317* (2021).
- [82] Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 1458–1467.
- [83] Jie Zou, Mohammad Aliannejadi, Evangelos Kanoulas, Maria Soledad Pera, and Yiqun Liu. 2022. Users Meet Clarifying Questions: Toward a Better Understanding of User Interactions for Search Clarification. *ACM Trans. Inf. Syst.* (mar 2022). <https://doi.org/10.1145/3524110>
- [84] Jie Zou, Yifan Chen, and Evangelos Kanoulas. 2020. Towards question-based recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 881–890.
- [85] Jie Zou and Evangelos Kanoulas. 2019. Learning to ask: Question-based sequential Bayesian product search. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 369–378.
- [86] Jie Zou and Evangelos Kanoulas. 2020. Towards Question-Based High-Recall Information Retrieval: Locating the Last Few Relevant Documents for Technology-Assisted Reviews. *ACM Transactions on Information Systems (TOIS)* 38, 3, Article 27 (May 2020), 35 pages.
- [87] Jie Zou, Evangelos Kanoulas, and Yiqun Liu. 2020. An Empirical Study on Clarifying Question-Based Systems. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2361–2364.
- [88] Jie Zou, Evangelos Kanoulas, Pengjie Ren, Zhaochun Ren, Aixin Sun, and Cheng Long. 2022. Improving Conversational Recommender Systems via Transformer-Based Sequential Modelling. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*. 2319–2324.